DM6810/DM6910 Driver for Windows 98/NT4/2000/XP

User's Manual
version 4.0

RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

SWM-640010005
Rev. B

ISO9001 and AS9100 Certified

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

08/19/2003	Revision A issued Documented for ISO9000
09/19/2003	Revision B issued Version 4.0 released Company name changed Windows 2000/XP support added Windows 95 support removed All function and files names *5810 changed to *6810 fixed a bug in the function RemoveIRQHandler6810 interface of the following function changed: - OpenBoard6810

DM6810/DM6910 Driver for Windows 98/NT4/2000/XP
Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

TABLE OF CONTENTS **4**

INTRODUCTION **5**

INSTALLATION **6**

INSTALLATION OF THE DRIVER AND EXAMPLE PROGRAMS 6
THE DM6810 BOARD DRIVER 7
THE DRIVER CONFIGURATION 7
INSTALLING A SECOND BOARD IN THE SYSTEM 8
WORKING WITH OTHER BOARDS USING THE WINRT DRIVER 8
USING WINDOWS REGISTRY KEYS 9

THE DRIVER API FUNCTIONS **11**

USING THE DRIVER WITHOUT THE WINDOWS-GUI **12**

DRIVER API FUNCTION GROUPS **13**

DRIVER INITIALIZATION FUNCTIONS 13
GENERAL BOARD CONTROL FUNCTIONS 13
DIGITAL I/O FUNCTIONS 13
INTERRUPT HANDLING 13
USER TIMER-COUNTER FUNCTIONS 13
ERROR HANDLING FUNCTIONS 14

ALPHABETICAL DRIVER API FUNCTIONS REFERENCE **15**

EXAMPLE PROGRAMS REFERENCE **29**

WIN32 WINDOWS APPLICATIONS 29

LIMITED WARRANTY **30**

Introduction

The DM6810/DM6910 board driver for Windows 98/NT4/2000/XP was designed for programmers who write Windows-based application programs for the RTD's DM6810 or DM6910 board. DM6810 and DM6910 are the same boards with different interface connector. Later we will refer both boards as DM6810.

The driver provides an Application Programming Interface with a lot of function calls to perform all the data acquisition tasks of the board users.

The board driver based on the BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are written in Microsoft Visual C++ ver. 6.0.

Installation

Installation of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6810 board in your PC. Please follow the instructions of the manufacturer, how to install the board in a computer.

To install the drivers for the DM6810 under Windows 98/2000/XP you need the installation diskette 0. The readme.txt file on this diskette describes the necessary installation steps.

For Windows NT4.0 the installation diskette 0 is not needed.

On the installation diskette 1 you can found the **Setup.exe** program, which installs on your PC the DM6810 board driver and example programs. The setup program automatically detects your operating system and installs the appropriate files on your PC.
After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.
The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'RTD Embedded Technologies' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6810 board, and the project files for Microsoft Visual C++ users to rebuild the programs. The example programs are compiled with the Microsoft Visual C++ ver. 6.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

Uninstallation: you can uninstall the DM6810 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6810 board driver

The RTD's DM6810 board driver handles the hardware through the BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's Drvr6810.dll provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers has multi-board feature. It means that it is possible to use more than one board in the system.

The base address of the DM6810 board can be set by DIP-switch. If you are using multiple boards in your system, you need assign a different base address to each board. The IRQ and DMA channels are set programmatically. When you install a WinRT driver for the board, you must select the IRQ and DMA channel, and during the operating of the board must be used this IRQ and DMA channels.

Each board requires a unique WinRT device to handle the hardware. The maximum number of devices is 32.

During the installation the Setup.exe program set the default driver values:
Board 1: 0x300 base address, and IRQ 11

If these settings are not proper to your PC configuration, you can modify the settings with the **Drvr_cfg.exe** program in Windows NT4.0.
In case of Windows 98/2000/XP the operating system uses a WDM driver and it can be configured in the Control Panel/System/[Hardware]/Device Manager.

The Driver Configuration

As in the previous chapter mentioned, after installing the driver the default settings are used.

If these settings are not proper to your PC configuration, or there are other RTD boards used, you can modify the settings.
Under Windows NT driver can be configured with the **Drvr_cfg.exe** program. This program is installed on your PC during the setup process and it can be started from the Start menu.

Program options:

- Device number: The number of device, which handles the board. Each board has a unique device number between 0 and 9.
- I/O port base address: The I/O base address of the board.
- IRQ number: The IRQ number using of the board.
- DMA channel: The DMA channel using of the board.
- DMA buffer size: The DMA buffer size. It should not be more than 64k.
- Register: Press this button to store the settings in the Windows registry.
- Refresh: Press this button to get the information from the registry about the selected device number.

Under Windows 98/2000/XP same changes can be made using Control Panel/System/[Hardware]/Device Manager/WinRT Devices.

Do not forget, that all boards are required different driver devices, and in your C++ program must be used the DEVICE_NO according to the driver device number.

Installing a second board in the system

It is possible to use more than one board in the system using the RTD's board drivers. Please follow the next steps to install a multi-board – system:

- Install the first board with all software and make sure that everything is working well.
- Place the second board in an empty slot in the PC, and switch on the computer.
  Windows NT4.0: After the system startup start the installation program and answer 'YES' to the next board installation question. Restarting the system with the two-board example program is possible to test the working of both boards. Please note that during the second board installation the driver and example program files will be refreshed thus you need to save the important changes before starting the install.
  Windows 98/2000/XP: You need only the installation diskette 0 to install the next boards. Follow the steps described in the readme.txt file on diskette 0.

Working with Other Boards Using the WinRT Driver

As you may have in your computer other RTD boards, which uses the WinRT driver, or a board from other manufacturer with WinRT, our installation program automatically detects the WinRT settings from the system registry file, and installs the appropriate (next) WinRT device. You can see an informational dialog box about this issue during the setup.
As an application written for the DM6810 board should know which WinRT device is assigned to the board, the installation program generates the file wrtdev.txt in the 'Examples' directory containing the actual device number. The example programs are reading this file at the startup to identify this number.
If the other board in your system, which uses the WinRT driver does not handle the situation when there is other WinRT-based application, install this board firstly, and then install the DM6810 which handles correctly this state.

Using Windows registry keys

The RTD board drivers are using the next registry keys:

Windows NT 4.0:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- WinRT
                    |
                    | -- WinRTdev0       (one per device)
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0

                    | -- WinRTdev1
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0
                    |
```

Windows 98 Enum key path:

```
HKEY_LOCAL_MACHINE
|
| -- Enum
```

Windows 98 Class key path:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- Class
                    |
                    | -- WinRTDevices
```

Windows 2000/XP device class information:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Control
               |
               | -- Class
                    |
                    | -- {D695ED6A-630D-4D83-D8B-F1F0AC107AD0}
                    |
                    | -- 0001
```

Windows 2000/XP Plug and Play devices are enumerated under the Enum key:

```
HKEY_LOCAL_MACHINE
|
| -- System
   |
   | -- CurrentControlSet
        |
        | -- Enum
             |
             | -- PCI
```

The Driver API Functions

The resources on the DM6810 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the Drvr6810.dll file. To write applications using the API functions you must include the Reg6810.H header file, and link the program with the Drvr6810.lib import library file.
In the example programs you can find different examples how to use the driver.

Using the Driver without the Windows-GUI

This driver is based on the Win32 system, and can run only under Windows. However, the user, who is not familiar with the Windows Graphical User Interface, can use the driver API functions in Win32 console applications, which has an MS-DOS like text-mode interface.

In console applications is not necessary to use the Windows graphical environment, but all the driver's API functions are accessible.
The Microsoft Visual C++ compiler supports writing console applications.

Driver API Function Groups

Driver Initialization Functions

OpenBoard6810
CloseBoard6810

General Board Control Functions

InitBoard6810

Digital I/O Functions

DIOClearChip6810
DIORead6810
DIOWrite6810
DIOSelectClock6810
DIOStrobeEnable6810
DIOIsChipStrobe6810
DIOSelectRegister6810
DIOSetPort0Direction6810
DIOSetPort1Direction6810

Interrupt Handling

ClearBoardIRQ6810
EnableBoardIRQ6810
EnableIrqSharing6810
InstallCallbackIRQHandler6810
InstallCounterIRQHandler6810
RemoveIRQHandler6810
GetIRQCounter6810
LoadIRQRegister6810
ReadBoardIRQStatus6810
SetIRQPolarity6810
IsEPLD1IRQ6810
IsEPLD2IRQ6810
IsEPLD3IRQ6810
IsP14IRQ6810
DIOClearIrq6810
DIOEnableIrq6810
DIOSetIRQSource6810
DIOIsChipIRQ6810

User Timer-Counter Functions

ClockDivisor6810
ClockMode6810
DoneTimer6810
ReadTimerCounter6810
SetUserClock6810

Error Handling Functions

GetErrorStatus6810

Alphabetical Driver API Functions Reference

C

ClearBoardIRQ6810

void ClearBoardIRQ6810(RTDHANDLE hBoard);

Description:

This routine is used to clear the board IRQ circuitry and status bit.
Parameters:

hBoard:	device handle

ClockDivisor6810

void ClockDivisor6810(RTDHANDLE hBoard, uchar8 Timer, uint16 Divisor);

Description:
This routine is used to set the divisor of a designated counter on the 8254 programmable interval timer (PIT). This procedure assumes that the counter has already been set to receive the least significant uchar8 (LSB) of the divisor followed by the most significant uchar8 (MSB).

Parameters:

hBoard:	device handle
Timer:	0,1,2
Divisor:	0 - 65535

ClockMode6810

void ClockMode6810(RTDHANDLE hBoard, uchar8 Timer, uchar8 Mode);

Description:

This routine is used to set the mode of a designated counter in the 8254 programmable interval timer (PIT).

Parameters:

hBoard:	device handle
Timer:	0,1,2
Mode:	0,1,2,3,4,5

CloseBoard6810

BOOL CloseBoard6810 (LONG DeviceNumber, LPSTR szBuf);

Description:

This routine is used to close board driver.

Parameters:

DeviceNumber:	WinRT device number to load
szBuf:	message buffer

Return value:

TRUE on success, FALSE on error

DIOClearChip6810

void DIOClearChip6810(RTDHANDLE hBoard, uchar8 Chip);

Description:

This routine clears the selected DIO chip.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)

DIOClearIRQ6810

void DIOClearIrq6810(RTDHANDLE hBoard, uchar8 Chip);

Description:

This routine clears the selected DIO chips IRQ status bit.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)

DIOEnableIRQ6810

void DIOEnableIrq6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Enable);

Description:

This routine enables the selected DIO chips interrupt.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
Enable:	0 = Disabled 1 = Enabled

DIOIsChipIRQ6810

uchar8 DIOIsChipIrq6810(RTDHANDLE hBoard, uchar8 Chip);

Description:

This routine checks to see if the selected DIO chip has generated an interrupt.

Parameters:

hBoard: device handle
Chip: 0, 1, 2 (selects which DIO chip on board)

Returns:

1 if IRQ has been generated
0 if no IRQ

DIOIsChipStrobe6810

---

uchar8 DIOIsChipStrobe6810(RTDHANDLE hBoard, uchar8 Chip);

Description:

This routine checks to see if the selected DIO chip has received strobe input.

Parameters:

hBoard: device handle
Chip: 0, 1, 2 (selects which DIO chip on board)

Returns:

1 if strobe has been received
0 if no strobe

DIORead6810

---

uchar8 DIORead6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Port);

Description:

This routine reads the selected DIO chips port.

Parameters:

hBoard: device handle
Chip: 0, 1, 2 (selects which DIO chip on board)
Port: 0, 1

Returns:

8 bit unsigned character value

DIOSelectClock6810

---

void DIOSelectClock6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Clock);

Description:

This routine picks the different registers on the selected DIO chip.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
Clock:	0 = 8 MHz
	1 = Programmable clock

DIOSelectRegister6810

void DIOSelectRegister6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Select);

Description:

This routine picks the different registers on the selected DIO chip.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
Select:	0 = Clear mode
	1 = Port 0 direction
	2 = Port 1 direction
	3 = IRQ source

DIOSetIRQSource6810

void DIOSetIRQSource6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 IRQSource);

Description:

This routine sets the selected DIO chips interrupt source.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
IRQSource:	0 - 15 (P0.0 - P1.7)

DIOSetPort0Direction6810/DIOSetPort1Direction6810

void DIOSetPort0Direction6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Direction);
void DIOSetPort1Direction6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Direction);

Description:

This routine sets the selected DIO chips port 0/1 direction.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
Direction:	0 - 255
	0 = In
	1 = Out

DIOStrobeEnable6810

void DIOStrobeEnable6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Enable);

Description:

This routine enables the strobe input of the selected DIO chip.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)
Enable:	0 = Disabled
	1 = Enabled

DIOWrite6810

void DIOWrite6810(RTDHANDLE hBoard, uchar8 Chip, uchar8 Port, uchar8 Data);

Description:

This routine writes the selected DIO chips port.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2
Port:	0, 1
Data:	0 - 255

DoneTimer6810

void DoneTimer6810(RTDHANDLE hBoard);

Description:

Initialize the timers for high speed to ensure the immediate load.

Parameters:

hBoard:	device handle

EnableBoardIRQ6810

void EnableBoardIRQ6810(RTDHANDLE hBoard, uchar8 Enable);

Description:

This routine enables the board IRQ for the DM6810.

Parameters:

hBoard:	device handle
Enable:	0 = disabled
	1 = enabled

EnableIrqSharing6810

void EnableIRQSharing6810(RTDHANDLE hBoard, uchar8 Enable);

Description:

This routine enables and disables the interrupt sharing feature of the board.

Parameters:

hBoard:	device handle
Enable:	1 = Disabled
	0 = Enabled

GetErrorStatus6810

BOOL GetErrorStatus6810 (RTDHANDLE hBoard, LONG *ErrorCode, LPSTR ErrorString);

Description:

This routine is used to get the error code and string in case of errors.
The function clears the last error code.

Parameters:

hBoard:	device handle
ErrorCode:	Error Code
ErrorString:	Return error string

Return Value:

TRUE	if error occurred
FALSE	otherwise

GetIRQCounter6810

void GetIRQCounter6810(RTDHANDLE hBoard, IRQSetup *irqsetup);

Description:

This routine is used to get the counter incremented by IRQ.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

InitBoard6810

void InitBoard6810(RTDHANDLE hBoard);

Description:

This routine should always be called first.
This clears the board and variables the driver uses.

Parameters:

hBoard:	device handle

InstallCallbackIRQHandler6810

BOOL InstallCallbackIRQHandler6810 (RTDHANDLE hBoard, isr_t handler);

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
isrT handler:	user callback function

Return value:

TRUE on success, FALSE on error

InstallCounterIRQHandler6810

BOOL InstallCounterIRQHandler6810(RTDHANDLE hBoard, IRQSetup *irqsetup);

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

Return value:

TRUE on success, FALSE on error

IsEPLD1IRQ6810/IsEPLD2IRQ6810/IsEPLD3IRQ6810

uchar8 IsEPLD1IRQ6810(RTDHANDLE hBoard);
uchar8 IsEPLD2IRQ6810(RTDHANDLE hBoard);

uchar8 IsEPLD3IRQ6810(RTDHANDLE hBoard);

Description:

This routine checks the EPLD1/EPLD2/EPLD3 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsP14IRQ6810

---

uchar8 IsP14IRQ6810(RTDHANDLE hBoard);

Description:

This routine checks the P14 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

LoadIRQRegister6810

void LoadIRQRegister6810(RTDHANDLE hBoard, uchar8 Value);

Description:

This routine loads the interrupt register with one write operation.

Parameters:

hBoard:	device handle
Value:	0 - 7

OpenBoard6810

BOOL OpenBoard6810(int num, int device_id, LPSTR szBuf, BoardConfig *boardconfig);

Description:

This routine is used to open board driver.

Parameters:

num:	the number of the board (a system can have more board)
device_id:	0x6810
szBuf:	return string
BoardConfig:	pointer to return board configuration

Return value:

TRUE on success, FALSE on error

ReadBoardIRQStatus6810

uchar8 ReadBoardIRQStatus6810(RTDHANDLE hBoard);

Description:

This routine returns the IRQ status from the board.

Parameters:

hBoard: device handle

Returns:

8 bit unsigned integer

ReadTimerCounter6810

uint16 ReadTimerCounter6810(RTDHANDLE hBoard, uchar8 Clock);

Description:

This routine is used to read the contents of the selected timer/counter.

Parameters:

hBoard: device handle
Clock: 0, 1, 2

Returns:
16-bit count value.

RemoveIRQHandler6810

void RemoveIRQHandler6810(RTDHANDLE hBoard);

Description:

This routine is used to uninstall IRQ handler function.

Parameters:

hBoard: device handle

SetIRQPolarity6810

void SetIRQPolarity6810(RTDHANDLE hBoard, uchar8 Polarity);

Description:

This routine sets the P14 interrupt to use the positive edge or negative edge.

Parameters:

hBoard:	device handle
Polarity:	0 = positive edge 1 = negative edge

SetUserClock6810

float32 SetUserClock6810(RTDHANDLE hBoard, uchar8 Timer, float32 InputRate,
float32 OutputRate);

Description:

This routine sets the user timer counter.

Parameters:

hBoard:	device handle
Timer:	0, 1, 2
InputRate:	Input clock to selected timer.
OutputRate:	Desired output rate from selected timer.

Returns:

The actual clock frequency that is programmed.

Example Programs Reference

Win32 Windows applications

Name	Feature	Remarks
Digital	Sample program that demonstrates how to program digital I/O.	Sample program that demonstrates how to read and write the digital I/O. The program sets Ports 0 and 1 as outputs. The rest of the ports are set as inputs. The program constantly displays the value of the lines on the input port in the center of the screen. You can enter a value to be sent to the output port. Wiring the input port lines to the corresponding output port lines will allow you to change the output lines and view the changes at the same time.
Digital_IT	Sample program that demonstrates how to use the P1.0 input as an interrupt source and how to write an Interrupt Service Routine (ISR).	Sample program that demonstrates how to read and write the digital I/O using interrupts. The program sets Port 0 as output. The rest of the ports are set as inputs. This program sets up the P2.0 input to generate interrupts. An interrupt service routine (ISR) is provided that simply sends a value out the output ports and reads a value from the input ports. The values are displayed on the screen as a foreground process. Although the ISR in this program is trivial, all the ingredients for powerful interrupt driven data acquisition are included. The timer is programmed to run at the specified rate. If you connect the output from timer/counter 2 to P2.0, interrupts will be generated at the timer rate.
Timer_IT	Sample program that demonstrates how to use the timer as an interrupt source and how to write an Interrupt Service Routine (ISR).	Sample program that demonstrates how to read and write the digital I/O using interrupts. The program sets Ports 0 and 1 as outputs. The rest of the ports are set as inputs. This program sets up the User Timer to generate interrupts at a selected rate. An interrupt service routine (ISR) is provided that simply sends a value out the output ports and reads a value from the input ports. The values are displayed on the screen as a foreground process. Although the ISR in this program is trivial, all the ingredients for powerful interrupt driven data acquisition are included.
Timers	Sample program that demonstrates how to program the user timer.	Sample program demonstrating how to program the 8254 programmable interval timers (PITs). Counter 0 is programmed to run at 200 Hz. Counter 1 is used to count these pulses. The result is converted into seconds and printed on the screen.

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com